Early Infusion of New Planning, Scheduling and Execution Technology into A Flight Mission

James Kurien[*] and Mark Drummond[†]
NASA Ames Research Center, Moffett Field, CA, 94034

Anthony Barrett[§] and Sven Grenander[‡]
Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91109

This extended abstract describes efforts to infuse new planning and scheduling technologies into the Mars Science Laboratory Mission (MSL), a NASA Mars rover mission planned for launch in 2009. Beginning in 2003, we engaged the MSL mission and the developers of the Mission Data System (MDS). MDS is a software system that at the time was the MSL software baseline for both the ground and flight system. We briefly describe the tools that we integrated with MDS, the analysis or experience on previous missions that suggested each tool, and our successes in integrating these tools into a proof-of-concept uplink system we demonstrated in late 2004. In 2004, MSL decided to fall back on the very successful Mars Exploration Rover (MER) mission's software in order to save development cost, which has resulted in some re-direction of our on-going activities. We briefly describe our new work to enhance the MER-based MSL software.

I. Introduction

Infusion of new software technology into flight projects has historically been quite difficult due to the conservative nature of flight project management. The conservative management stance is driven by the need to succeed in satisfying the missions' stated objectives under strict cost and schedule constraints. Traditionally, missions have not had the luxury of being software technology trailblazers except where some unique mission requirement has forced new technology to be adopted. The task described in this report was put in place to try to overcome the software technology infusion hurdle by placing the software technologists in direct contact with a real mission and to a large extent, at the disposal of the mission.

The Planning, Scheduling and Execution (PS&E) Technology Infusion task has as its charter to do technology identification, development and adaptation specifically to infuse PS&E technologies developed under "Intelligent Systems" funding into the Mars Science Laboratory mission. *Intelligent Systems* is a funding program that now lives within NASA's Exploration Systems Mission Directorate (ESMD), and it has invested in a variety of autonomy technologies that should be useful to a number of human and robotic exploration missions. The personnel on the task are PS&E technologists who are working within the project on a daily basis to anticipate mission needs and prepare selected IS developed PS&E technologies so that they can be made available to the MSL mission without risking either MSL mission development or operations costs or schedules. The task answers to mission PS&E needs first and foremost, providing solutions only to identified problems, not pushing any specific technology or technology agenda. It is in the very nature of early project phases to go through major design fluctuations as the project explores a huge trade-space to find the best combination of tools, procedures, technologies *etc.* to meet the mission objectives. This task has been part of that trade-space exploration, both helping identify trades and having to adjust to changes in plans as trade-space options were chosen or discarded.

The rest of this paper describes the current state of the PS&E technology evaluation, selection and infusion process. The paper shows that the infusion team has been able to stay committed to its charter and has had a positive influence on the mission and is poised to stay with the mission through development and hand-off into flight. It is expected that this task will serve as a model for future well considered technology infusion tasks and that it will at

[*] NASA Ames Principal Investigator, Computational Sciences Division, MS 269-4.
[†] NASA Ames Task Manager, Computational Sciences Division, MS 269-1.
[§] JPL Principal Investigator, Planning and Execution Section, MS 126-347.
[‡] JPL Task Manager, Planning and Execution Section, MS 301-285.

least partially address what has been a long standing problem in NASA software technology adoption for mission use.

II. The Infused Technologies

Figure 1 illustrates the tools we chose to integrate with MDS. It also illustrates the relationship of these tools to MDS' abilities to generate a flexible plan from a set of goals (as indicated by the Ground MDS box) and to execute a flexible plan on-board a rover (as indicated by the Flight MDS box). A goal editor allows the user to graphically choreograph high level rover activities such as driving and picture taking. The goals for the rover are then supplied to MDS, which elaborates them into a more detailed plan for coordinating the various subsystems of the rover to achieve the goals. This plan is temporally flexible, in that it need not specify fixed times for each activity in the plan. Instead, it contains more general specifications of how the activities in a plan should be ordered, such as legal time ranges for the start of an activity or relationships between the start and end times of different activities. Before sending the flexible plan to the instance of MDS operating on the rover, we must determine if the plan is an acceptable implementation of our goals for the rover. To assist in this determination, we first integrated a timeline visualization tool. This allows operators to view the flexible plan as a traditional fixed-time sequence of activities. In order to enable operators to gain an understanding of how the plan was likely to behave given uncertainty about the rover and its environment, we integrated a Monte Carlo evaluation system that automatically simulates and summarizes a large number of executions of the plan under a wide variety of possible operating conditions. If the plan generated by MDS was not accepted by the operator, the first course of action is to adjust the goals given to MDS. This puts the user in the situation, common to all automated planning systems, of having to determine why the original goals did not result in a desirable plan. We therefore integrated an inspection system that allows the user to explore how MDS generated a flexible plan from the goals and why some more desirable plan was not produced. Each of these tools is described in more detail below. We also performed significant computational benchmarking of the MDS system, which is not discussed further here (but details are available upon request from the authors).

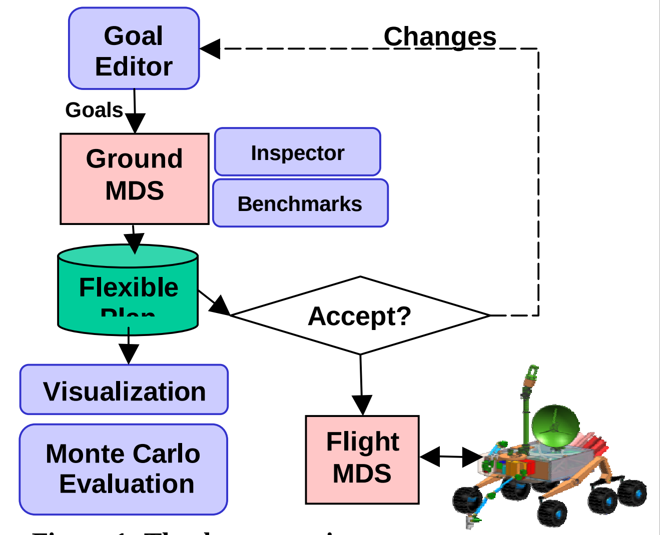
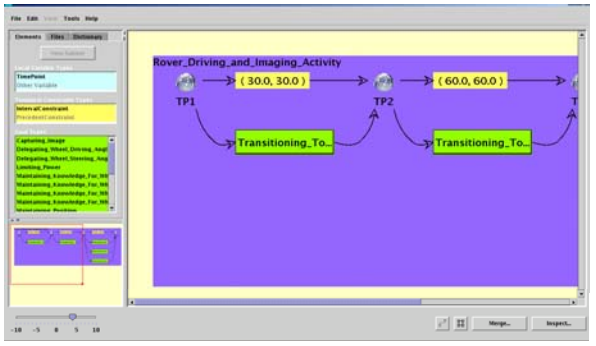


Figure 1: The demonstration system

A. Goal Editor

Figure 2 shows the goal editor, which allows the user to graphically select and choreograph high level rover activities. Operation of a spacecraft at JPL typically involves an activity dictionary, which defines the validated activities that an operator may perform with the spacecraft. The left-most panel of the goal editor contains a list of parameterized activities that may be selected for the rover. In this example, they are high level goals such as transitioning to a new position relative to rover's current position, or taking an image. Parameters include the position to be achieved and which of several cameras to be used, respectively. The right side of the editor is a canvas upon which activities are arranged. Here we see two *Transitioning_To* goals that will move the rover. Rather than specifying the


File Edit View Tools Help
Elements Files Dictionary
TimePoint
IntervalConstraint
Capturing_Image
Delegating_Wheel_Driving_Angl
Delegating_Wheel_Steering_Ang
Limiting_Power
Maintaining_Knowledge_For_Wh
Rover_Driving_and_Imaging_Activity
(30.0, 30.0)
(60.0, 60.0)
TP1
TP2
Transitioning_To...
Transitioning_To...
-10 -5 0 5 10
Merge...
Inspect...
Figure 2: The goal editor

exact times that each movement will begin, and risk the plan failing if the first drive takes longer than expected, MDS allows us to specify the timing or simply the ordering of activities with constraints. To ensure the second movement starts as soon as the first completes, but no sooner, we constrain the second *Transitioning_To* goal to begin at the same point in time where the first *Transitioning_To* ends. On the editor canvas, this common point in time is the circle labeled TP2, and the arrows around TP2 represent the constraints on the goal start and end times. The goal editor allows us to graphically specify and visualize temporal constraints between the activities we have chosen for the rover. The editor then submits our goals to MDS. MDS elaborates each high level goal, such as transitioning the rover to a new position, into a set of lower-level activities that cause the rover to implement that goal. This elaboration process results in a much larger and more detailed set of activities and a much larger set of temporal constraints specifying the legal times at which each may start.

B. Timeline visualization

Ensuring that an elaborated set of goals will execute as human operators intend involves inspecting those goals using a timeline visualizer like the one shown in Figure 3. From top to bottom, this visualizer consists of a header with a time scale, a set of intervals associated with elaborated goals, and a set of constraints on different state variables representing spacecraft features. This GUI was adapted from the ASPEN[1] display in order to support visualization of arbitrary constraints from the grayed-out unconstrained variable to the textual description and the magenta lines denoting constraints to maintain knowledge of a state variable's value.

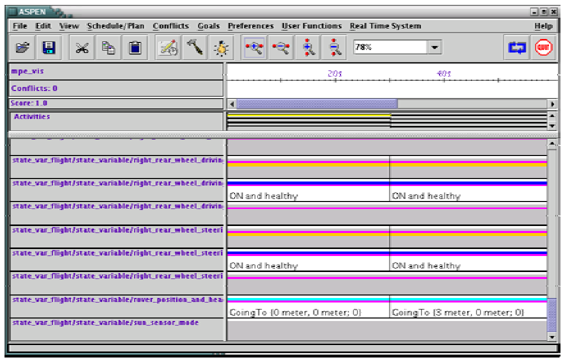

ASPEN
File
Edit
View
Schedule/Plan
Conflicts
Goals
Preferences
User Functions
Real Time System
Help
78%
mpe_vis
Conflicts: 0
Score: 1.0
Activities
20s
40s
state_var_flight/state_variable/right_rear_wheel_drivin
state_var_flight/state_variable/right_rear_wheel_drivin
ON and healthy
ON and healthy
state_var_flight/state_variable/right_rear_wheel_drivin
state_var_flight/state_variable/right_rear_wheel_steeri
state_var_flight/state_variable/right_rear_wheel_steeri
ON and healthy
ON and healthy
state_var_flight/state_variable/right_rear_wheel_steeri
state_var_flight/state_variable/rover_position_and_hea
GoingTo (0 meter, 0 meter; 0)
GoingTo (3 meter, 0 meter; 0)
state_var_flight/state_variable/sun_sensor_mode

Figure 3: The timeline visualizer

For instance, the bottom row denotes constraints on a sun sensor state variable. Since neither of the *Transitioning_To* goals elaborates to add a constraint on the sun sensor, this state variable has no constraints – resulting in it being completely grayed-out. The next row up denotes the *position&heading* state variable, which has the original two *Transitioning_To* goals. These goals elaborate into simultaneous goals on the rover's 6 wheels, and the rows displayed above the *position&heading* state variable denote the low-level goals on the right rear wheel driving and steering motors. Finally, for display purposes the GUI shows the earliest possible times that given constraints can start holding.

C. Monte Carlo Evaluation

The timeline visualization of a plan allows us to view one possible execution of a temporally flexible plan. However, due to the flexible nature of the plan and the inherent uncertainty about the conditions under which it will execute, we don't know exactly how it will cause a rover to behave on Mars. Monte Carlo evaluation of a temporally flexible plan helps us to estimate what the likely or worst case behavior of a plan will be, given some bounds on the conditions under which it will execute. We believe this will allow rover operators to better understand how a plan is likely to behave, and to adjust it accordingly or explore alternatives.

Consider the simple scenario shown in Figure 4. As shown in the images in the top of the figure, we would like the rover to turn 90 degrees, drive forward 3 meters, and take a series of images. The graph below indicates the amount of energy in the rover's batteries as execution of the plan progresses. This graph is meant to be suggestive rather than indicative of real values from any specific rover mission. The initial energy stored in the battery comes from allowing the rover to nap, that is to not perform any

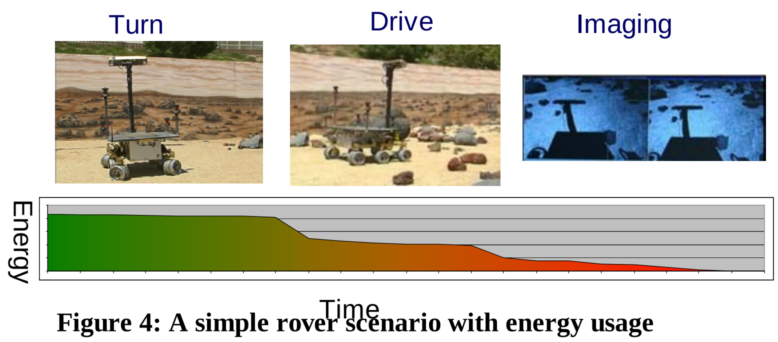


Figure 4: A simple rover scenario with energy usage

activities, while the solar panels charge the batteries. The energy level at the end of the plan is determined by factors such as how long the drive takes and the power usage over time by the wheel motors on the particular terrain the rover encounters. The energy level must remain above some safe threshold during execution of the plan, or the plan will be aborted.

Suppose we would like to know how much energy the rover should store up during its nap in order to ensure that the plan completes with a safe amount of energy in reserve. We might also ask, if our estimates of how much power the wheel motors will require is off by 10%, does that significantly impact the likelihood the plan will succeed? Since scientists will be trying to squeeze as many competing science tasks onto the rover as possible, we may also ask about the abort probability of the last image we plan to take. This image-taking activity may be aborted for lack of energy if we squeeze the rover's napping period a small amount in order to accommodate other actions earlier in the day.

The graph in Figure 5 illustrates how Monte Carlo simulation helps us to answer these types of analysis questions. The graph shows the initial energy stored before executing the simple scenario versus the estimated percentage of plans run with that initial energy that would complete their execution. Each column was generated by running 200 simulations of the plan with the specified initial energy. Each run chooses duration and power usage for each activity within a 10% standard deviation. This is the basis of all Monte Carlo simulation systems: analysis of multiple simulations of a system with variations chosen from a statistical model of the system's behavior. We can use these simulations not only to estimate how likely a plan is to complete, but to record where in its execution each simulated run of the plan fails. This allows us to explore, for example, how many of the images at the end of this plan are likely to be taken given a certain energy budget.

The particular Monte Carlo simulation system we have integrated with MDS was originally a component of the PICo[2] planning system. In addition to temporal and power variations, it can simulate other resources and inject failures into the simulated runs. Through a file format adapter, it reads in a plan generated by MDS and converts it into a high level simulation. This is not a physical simulation of the rover. Conceptually, the simulation bookkeeps which activities must be completed before subsequent activities start, and how much energy has been used. Nominal values for activity durations, power usage and the like can specified by engineers, determined from a high level simulation, or even measured on a test bed. The range of variation in these parameters that's of interest for simulation is then provided by the user.

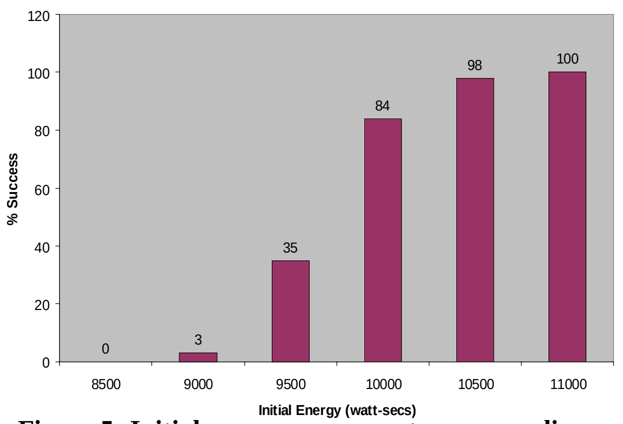


Figure 5: Initial energy vs. percentage succeeding

D. PlanWorks

PlanWorks[3] allows a software developer or rover operator to understand how a flexible plan was generated by the planner within MDS, or other planners. It can also reveal why a planner was unable to develop a plan for a set of goals. PlanWorks captures the planning and scheduling decisions made by the planner, such as which lower-level activities a high level goal will be decomposed into, and what temporal constraints or fixed execution times will be assigned to those activities. All of this information is then stored in an SQL database which can be queried to understand the process by which the planner developed the plan. PlanWorks also supplies a set of specialized tools for graphically exploring the planning process. For example, if the elaboration process in MDS was unable to generate a plan, one could view all of the goals that attempt to control the same property of the rover, such as its position. This would allow us to see, for example, if a mistake in our high level goals forced a drive activity of the rover to overlap with the stationary picture taking activity, which is not allowed. We can use PlanWorks during development time to investigate whether a planner is implemented correctly, or during operations to determine whether the goals we have set forth are correct for the plan we are expecting to receive in return.

III. Summary and Conclusion

In 2004, we created a connection to the MSL mission and the MDS software development team. We demonstrated that it is possible to infuse new technology into a mission's software baseline in a way that is minimally disruptive and quickly yields a proof of concept demonstration. Our demonstration illustrates four different views of the mission tactical activity planning process. These views focus on high level goals, flexible plans viewed as sequences, flexible plans viewed as descriptions of likely behavior, and a database of planner decisions, respectively. In this demo assembled over the course of six months, each of these views is implemented by a separate tool which is not well integrated with the others. To deploy these tools in an operational context, we would work to re-factor these tools into a set of components that can work together, and which interact with MDS through a unified API.

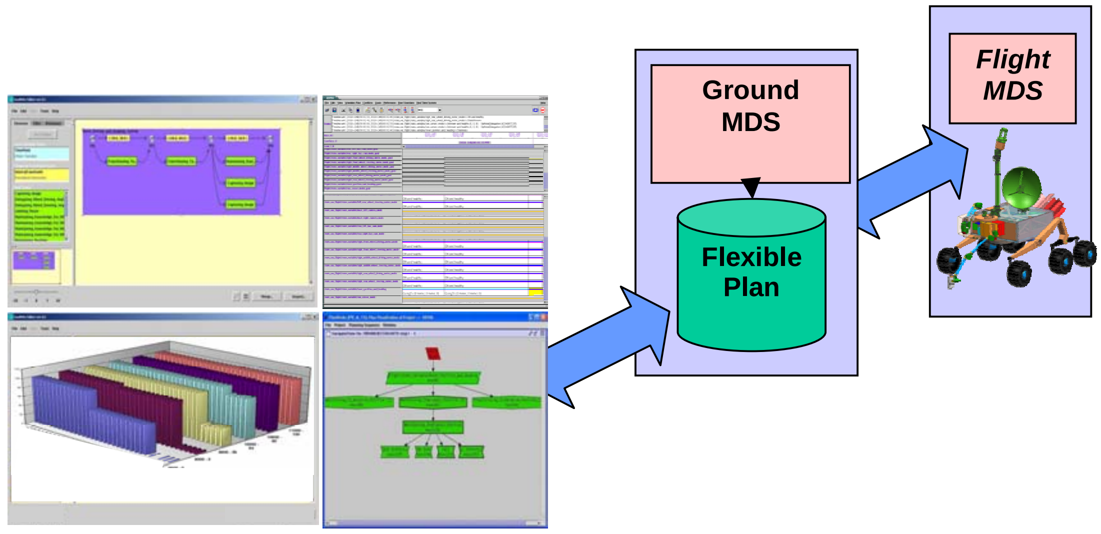


Figure 6: A more integrated environment for operations.

As 2005 approaches, MSL has adopted the software from the very successful MER mission as its baseline. The MER planning, sequence generation and uplink software contains functionality similar to that described above. It includes tools that perform automated or semi-automated planning, editors that allow users to specify temporal constraints between, and so on. To an even greater extent than the MDS demonstration described here, the MER uplink software consists of many tools, resulting in multiple user interfaces for similar tasks, replicated functionality across tools, and multiple representations for the same information about the rover or the plan.

The MSL mission's software priority is containment of development costs, followed by capability improvements that focus on cost or risk reduction during development or mission operations. Given our charter to maintain an MSL mission focus, our task for 2005 is to aid the mission in unifying the many MER tools, interfaces and models to reduce the cost of developing and maintaining an MSL-adapted version of the software. The MER uplink system has done an excellent job. With the benefit of additional time, lessons learned, and some techniques from the Intelligent Systems program, we will work to demonstrate that the MER uplink software can be affordably and incrementally re-factored in a way that allows tools to share the same models, UI components, and implementation of common functionality. We hope to demonstrate that the result can be lower costs for adapting the system to the specifics of the MSL mission, and a greater ability to add or modify features late in development of the mission, as is usually required. We will work with teams both inside and outside MSL that are maintaining or revising the tools used in the MER uplink system to develop an integrated, proof-of-concept demo in 2005.

Acknowledgments

This project was funded by the NASA Intelligent Systems program, now part of the Exploration Systems Mission Directorate at NASA Headquarters. In addition to the authors, this work was performed by George Stebbins at JPL, Kevin Greene and David Roland at NASA Ames Research Center, and Othar Hansson and Jordan Hayes of Thinkbank. The goal editor was developed by Alan Tomotsugu and Ted Shab at NASA Ames Research Center. This project could not have even begun without the assistance of the MDS development team. We especially thank Matthew Bennett, Michel Ingham, Robert Keller, Russell Knight, Richard Morris, Alex Murray, Robert Rasmussen, Mohammad Shahabuddin, I-hsiang Shu, Lin Song, and David Wagner. We would also like to acknowledge the support of the MSL project.

References

[1] S. Chien, G. Rabideau, R. Knight, R. Sherwood, B. Engelhardt, D. Mutz, T. Estlin, B. Smith, F. Fisher, T. Barrett, G. Stebbins, D. Tran , "ASPEN - Automating Space Mission Operations using Automated Planning and Scheduling," *SpaceOps 2000*, Toulouse, France, June 2000.

[2] Bresina, J., Dearden, R., Meuleau, N. Ramakrishnan, S., Smith, D. E., and Washington, R., “Planning under continuous time and uncertainty: A challenge for AI,” *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, San Francisco, CA, 2002, pp. 77-94.

[3] PlanWorks, Software Package, Ver. Milestone 15. NASA Ames Research Center Computational Sciences Division, Moffett Field, CA, 2004.

.